In the Specification

Page 1, before the first line, insert:

This is a continuation of application Serial Number 10/114,652 filed on April 1, 2002 which is a continuation of application Serial Number 09/238,446 filed on January 28, 1999, now U.S. Patent No. 6,366,999, each of which is incorporated by reference herein in its entirety.

Replace the paragraph beginning at page 6, line 10, with the following rewritten paragraph:

It is desirable to have an efficient mechanism ~~for~~ or means to generate complex conditions in each PE that can be specified and tested for by conditional instructions. This has the effect of changing SP conditional branches into PE data dependent execution operations. Having an effective means for parallel array conditional execution minimizes the need to have the PEs send condition signals back to the controller, which takes time and implementation expense, for the purposes of supporting conditional branching based on PE conditions. An implication of having parallel array conditional execution is that the approach chosen for providing PE condition feedback to the array controller can be simple in nature and less costly than providing condition signaling paths from each PE. By saving the condition flags in a programmer accessible register space that can be copied or moved to a PE's register file the flags can be easily communicated between PEs. In conjunction with a merged SP/PE as described more fully in U.S. Application Serial No. 09/169,072 filed October 9, 1998 entitled Methods and Apparatus for Dynamically Merging an Array Controller with an Array Processor Element, flags saved in PE0 are easily transferred to the SP. Using a log N reduction method, where N is the number of PEs in the array, it is possible to exchange PE flag information between all PEs in log

N steps. The transfer of condition information is consistent with the design of the existing ManArray network and does not require the addition of condition signaling paths between the PEs and the SP controller.

Replace the paragraph beginning at page 8, line 9, with the following rewritten paragraph:

Further details of a presently preferred ManArray architecture for use in conjunction with the present invention are found in U.S. Patent Application Serial No. 08/885,310 filed June 30, 1997, U.S. Patent Application Serial No. 08/949,122 filed October 10, 1997, U.S. Patent Application Serial No. 09/169,255 filed October 9, 1998, U.S. Patent Application Serial No. 09/169,256 filed October 9, 1998, U.S. Patent Application Serial No. 09/169,072 filed October 9, 1998, U.S. Patent Application Serial No. 09/187,539 filed November 6, 1998, U.S. Patent Application Serial No. 09/205,558 filed December 4, 1998, U.S. Patent Application Serial No. 09/215,081 filed December 18, 1998 and U.S. Patent Application Serial No. [[_____]] 09/228,374 filed January 12, 1999 and entitled "Methods and Apparatus to Dynamically Reconfigure the Instruction Pipeline of an Indirect Very Long Instruction Word Scalable Processor", Provisional Application Serial No. 60/071,248 entitled "Methods and Apparatus to Dynamically Expand the Instruction Pipeline of a Very Long Instruction Word Processor" filed January 12, 1998, Provisional Application Serial No. 60/072,915 entitled "Methods and Apparatus to Support Conditional Execution in a VLIW-Based Array Processor with Subword Execution" filed January 28, 1998, Provisional Application Serial No. 60/077,766 entitled "Register File Indexing Methods and Apparatus for Providing Indirect Control of Register in a VLIW Processor" filed March 12, 1998, Provisional Application Serial No. 60/092,130 entitled

"Methods and Apparatus for Instruction Addressing in Indirect VLIW Processors" filed July 9, 1998, Provisional Application Serial No. 60/103,712 entitled "Efficient Complex Multiplication and Fast Fourier Transform (FFT) Implementation on the ManArray" filed October 9, 1998, Provisional Application Serial No. 60/106,867 entitled "Methods and Apparatus for Improved Motion Estimation for Video Encoding" filed November 3, 1998, Provisional Application Serial No. 60/113,637 entitled "Methods and Apparatus for Providing Direct Memory Access (DMA) Engine" filed December 23, 1998 and Provisional Application Serial No. 60/113,555 entitled "Methods and Apparatus Providing Transfer Control" filed December 23, 1998, respectively, and incorporated by reference herein in their entirety.

Replace the paragraph beginning at page 14, line 9, with the following rewritten paragraph:

The architecture specifies an execute True, an execute False, an unconditional execute, and other specific operations on a per instruction basis, in a hierarchical fashion. The True and False flag settings are used for Execute if Condition is True and Execute if Condition is False operations. Branches may also occur based on the True or False condition state of these flags. For certain DSU instructions, Shift/Rotates, PEXCHG (a communication instruction), and selected ~~others~~ other instructions, the ManArray architecture provides the ability to specify how to update the ACFs using one of the four scalar conditions C, V, N, or Z side effects on an instruction by instruction basis. When executing VLIW operations, the programmer must select which of the arithmetic units, presently only the ALU or DSU although it will be recognized this capability can be extended, controls the setting of the single set of flags. Each cycle, the setting of the condition flags is explicitly specified by the instruction.

Replace the paragraph beginning at page 20, line 16, with the following rewritten paragraph:

Comparison instructions are always executed and always affect the flags. General-purpose flags (ACFs) are affected based on the condition code specified as part of the comparison instruction. Using condition combination, the previous state of the flags can be combined with the result of the condition code test specified by the current compare instruction. This approach allows complex conditions to be created without resorting to multiple branching. In Fig. 2B, CC stands for a condition code 202 such as Greater Than (GT) 206, Less Than (LT) 208, Equal (EQ) 204, or Less Than or Equal (LEQ) 207. The Compare (CMPcc) instruction 200 in Fig. 2A specifies the desired conditions CC, Fig. 2B, to be tested, the two source registers to be compared, the data type covering packed forms, and a Boolean combination specification field labeled ~~CComb~~ CCombo.

Replace the paragraph beginning at page 2, line 22, with the following rewritten paragraph:

--In SMIMD, a different VLIW can exist at the same VIM address which can then be executed in parallel for purposes of optimizing performance in different applications with varying needs for VLIW parallelism. For SMIMD code, the programmer specifies which arithmetic unit affects the flags when the VLIW is loaded as part of the Load VLIW (LV) instruction. This approach allows different PEs to have different units affect the flags. For SIMD code, the programmer specifies which unit affects the flags at execution time as part of the XV instruction. The XV instruction specification may override the unit specified in the LV

instruction. This allows the programmer to pack multiple non-overlapping VLIWs in the same VIM address with different arithmetic units affecting the condition flags per VLIW execution.--

Replace the paragraph beginning at page 27, line 3, with the following rewritten paragraph:

--In the ManArray processor, the concept represented in Fig. 5A is extended to the VLIW architecture as shown in Fig. 5B. In Fig. 5B, a conditional execution VLIW unit 550 is shown containing three execution units, a Data Select Unit (DSU) 560, an ALU 570, and a Multiply ~~Add~~ Accumulate Unit (MAU) 580. This hardware is incorporated in the SP and in each PE of a ManArray processor such as processor 100 shown in Fig. 1. Internal to these units are the basic operative elements and their ASF generation and latch units namely in the DSU 560 functional unit *fn* 562, in the ALU adder 572, and in the MAU multiplier 582. Three types of flag functionality are shown to demonstrate the versatility of the concept and are representative of typical application needs. In the DSU 560, the ASF, (C N V Z), are generated as required by DSU instructions. The ACFs 561 are generated in ACF generation unit 568 based upon the ASFs 563 with no feedback of stored ACF state from the programmer visible latches 598. The ALU 570 maintains the functionality of the approach illustrated in Fig. 5A. In the ALU 570, the ACFs 571 are generated in ACF generation unit 578 based upon the ASFs 573 and stored ACF state 599 fed back from the programmer visible latches 598. The MAU 580 utilizes a relatively simple mechanism with no ACFs being generated as a result of an MAU instruction. The MAU is not precluded in general from setting the ACFs as shown by this exemplary implementation. Only the architecturally defined ASFs, (C N V Z), 587 for the least-significant operation of an MAU instruction that affects these flags, are sent to multiplexer 592 where if selected they would

pass through to multiplexer output 597 and be latched in programmer visible state latches 596. The multiplexer 592 selects the ASFs generated from the MAU 587, from the ALU 577, from the DSU 567, or from the CNVZ state latch 589 as controlled by the CNVZ mux control signal 591. For VLIW execution, the Unit Affecting Flags (UAF) field in the load VLIW (LV) instructions, a 2 bit field in the present ManArray architecture, in conjunction with the UAF of an XV instruction determines the multiplexer control signals 591 and ACF Mux Control 593 as follows. The LV instruction's Unit Affecting Flags (UAF) bits are used to select which arithmetic instruction slot (A=ALU, M=MAU, D=DSU) is allowed to set condition flags for the specified VLIW when it is executed. The XV instruction's Unit Affecting Flags (UAF) bits override the UAF specified for the VLIW when it was loaded via the LV instruction. The override selects which arithmetic instruction slot (A=ALU, M=MAU, D=DSU) or none (N=NONE) is allowed to set condition flags for this execution of the VLIW. The override does not affect the UAF setting specified via the LV instruction as these are loaded in the VIM at the specified VLIW address. In the instruction syntax, a flag parameter is used to specify the UAF for the instruction. A blank parameter, i.e. 'F=', selects the UAF specified when the VLIW was loaded to be used for the instruction execution and consequent control of the multiplexers 592 and 594 to load the proper flags into the programmer visible registers 596 and 598. For example, with the UAF indicating the MAU is to affect the flags, multiplexer 592 selects in response to CNVZ Mux Control 591 signal, path 587 to pass through to multiplexer output 597 to load the generated CNVZ ASFs to CNVZ state latches 596. Since the MAU does not generate any ACFs as shown in the exemplary MAU 580, no ACFs are to be latched into the programmer visible ACF State Latches 598 and they retain their previous state. The MAU may still conditionally execute based upon the ACF values generated by another execution unit 595 following its

pipeline sequence. If no instruction sequence requires the CNVZ or ACF state latches 596 and 598 respectively to be updated by any execution unit, then the mux control signals 591 and 593 cause the multiplexers 592 and 594 to select the state latch outputs 589 and 599 to pass through to their multiplexer outputs 597 and 595 respectively. For the ManArray implementation, the bus widths for the CNVZ and ACF signals are shown in Fig. 5B where the CNVZ paths 587, 577, and 567 are all 4 bit signals corresponding to the C, N, V, and Z values. The ACF paths 571 and 561 are each 8 bit signals corresponding to F7-F0. The outputs of the multiplexers 592 and 594 are the 4 bit signal and the 8 bit signals, respectively, both of which are used in the branch logic. The ACF multiplexer output signals 595 are used to control conditional execution in each of the execution units 560, 570, and 580. In the SP, only the multiplexer output signals 595 and 597 are used in the branch logic for conditional branch execution.--